



Performance Profiling with AMD GPU Tools: A Case Study

Holger Gruen
European Developer Relations
Holger.Gruen@amd.com

Agenda

- Introduction
- Motivation
- Introduction to AMD GPUs Tools
- Preconditions for profiling the GPU load of a game
- Case Study: Blue Bytes "The Settlers – 6" (working title)
- Conclusions
- Questions and Answers

Agenda

- Introduction
- Motivation
- Introduction to AMD GPUs Tools
- Preconditions for profiling the GPU load of a game
- Case Study: Blue Bytes "The Settlers – 6" (working title)
- Conclusions
- Questions and Answers

Introduction

- GPU bottlenecks are non-trivial to understand
 - All understanding derived from observations
 - And could be wrong
 - Even experienced developers can get it wrong 😊
- To rectify this AMD introduced two new GPU tools
 - GPU PerfStudio
 - Real-time performance monitoring
 - GPU ShaderAnalyzer
 - Accurate performance estimates for shaders
- Both tools are DX9 only at the moment
 - The tools talk did demo DX10 and OpenGL versions

Agenda

- Introduction
- Motivation
- Introduction to AMD GPUs Tools
- Preconditions for profiling the GPU load of a game
- Case Study: Blue Bytes "The Settlers – 6" (working title)
- Conclusions
- Questions and Answers

Motivation

- Show that AMD GPU tools are useful
 - They can be used on real games
 - They deliver insights that are hard to get otherwise
 - Case study: Blue Bytes ,The Settlers 6' (working title)
- Hope to get you use the tools
 - Try them out
 - Take advantage of their features
 - Give us feedback – we want to improve our tools

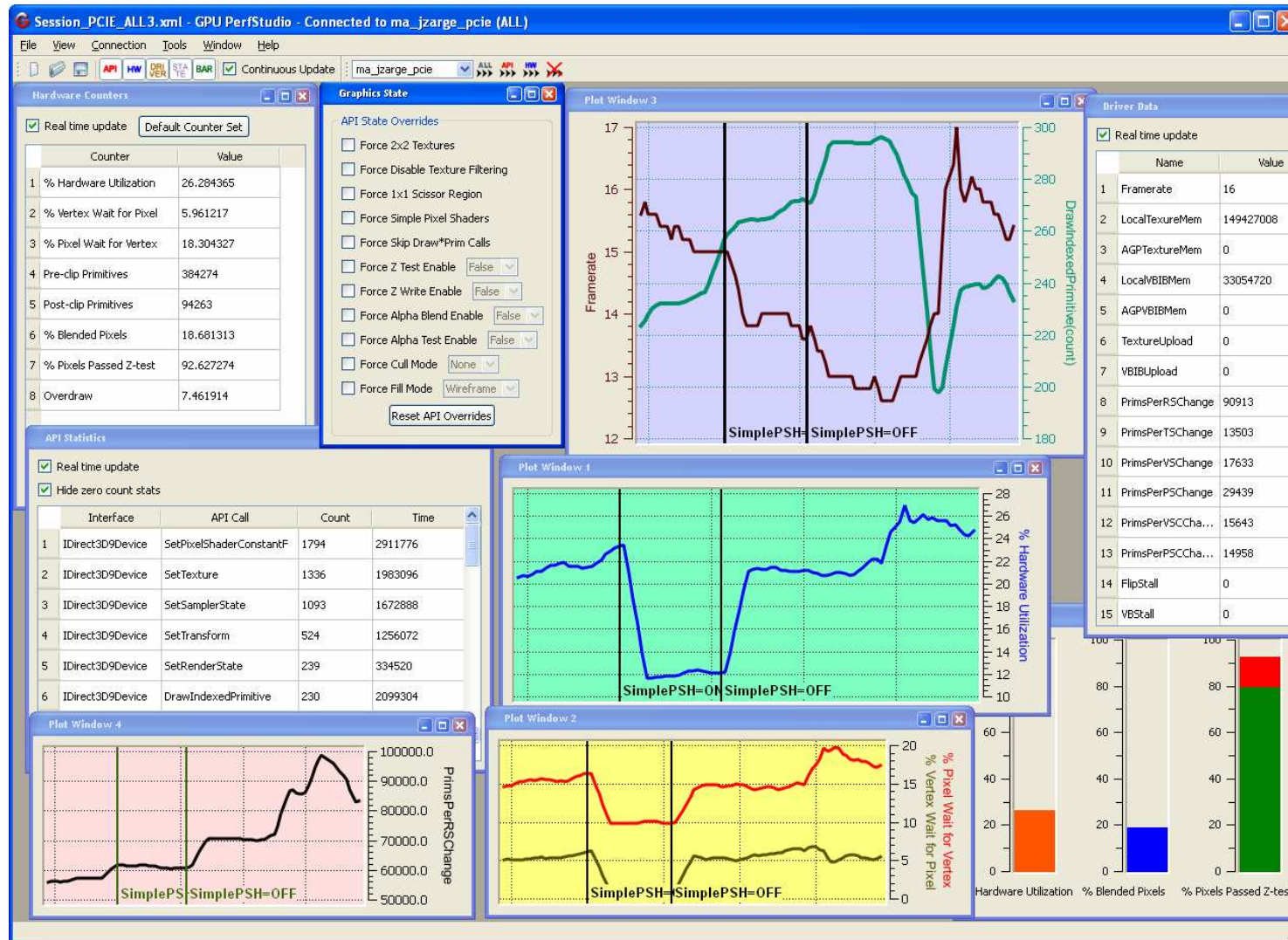
Agenda

- Introduction
- Motivation
- Introduction to AMD GPUs Tools
- Preconditions for profiling the GPU load of a game
- Case Study: Blue Bytes "The Settlers – 6" (working title)
- Conclusions
- Questions and Answers

Introduction to AMDs GPU tools

- GPU PerfStudio – real-time view window to the GPU
 - GPU hardware counters
 - Driver Data
 - API statistics
 - Graphics state overrides
 - Client/Server model
- GPU ShaderAnalyzer
 - Accurate performance estimates for shaders
 - Tells you about ALU:TEX
 - Lets you see the hardware shader
 - GUI and command line mode

Screenshot GPU PerfStudio



Screenshot GPU ShaderAnalyzer

The screenshot displays the AMD GPU ShaderAnalyzer application window. The interface is divided into several panes:

- Source Code:** Shows the HLSL source code for a pixel shader function named `ps_main`. The code includes comments and HLSL syntax for normalizing vectors and computing derivatives.
- HLSL Compiler:** Contains buttons for `Compile` and `Options`. The `Target` is set to `ps_3_0`. Checkboxes for `Avoid Flow Control` (unchecked) and `Prefer Flow Control` (checked) are visible, along with `Skip Optimization` (unchecked).
- Macro Definitions:** A table with columns for `Symbol` and `Value`. It currently contains the text "Right-click to add macros."
- Object Code:** Displays the compiled D3D Assembly code, showing instructions like `mov`, `break_ge`, `add`, `mad`, `texldd`, `if_lt`, and `endif`.
- Compiler Statistics (Using Catalyst 6.10):** A table showing performance metrics for various AMD GPUs.
- D3D Assembly Statistics:** Provides details about the shader, including the version, instruction count, and register usage.

Compiler Statistics (Using Catalyst 6.10)

	ASIC	Registers	Cycles (Bilinear)	Cycles (Trilinear)	Cycles (Aniso)	ALU:Tex (Bilinear)	ALU:Tex (Trilinear)	ALU:Tex (Aniso)
Radeon 9700 (R300)	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Radeon x800 (R420)	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Radeon x850 (R480)	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Radeon x1800 (R520)	12	5370.00	5370.00	5370.00	1.53	1.28	1.09	
Radeon x1900 (R580)	12	3506.25	4207.50	4908.75	0.53	0.45	0.38	

D3D Assembly Statistics

Shader Version = 3.0
 Instruction Count = 143
 ALU Instructions = 111, Texture Instructions = 11, ALU:Texture Ratio = 10.09

Constant Register Count = 3511
 Temp Register Count = 9, Sampler Register Count = 11, Input Register Count = 9, Output Register Count = 4

Agenda

- Introduction
- Motivation
- Introduction to AMD GPUs Tools
- Preconditions for profiling the GPU load of a game
- Case Study: Blue Bytes "The Settlers – 6" (working title)
- Conclusions
- Questions and Answers

Preconditions for profiling the GPU load of a game



- Make sure game logic is not in your way
 - Disable AI, physics ... the full simulation
- Profiling a full frame does not tell you all
 - Different techniques/passes have different bottlenecks
 - Need to be able to switch on/off selectively

Also:

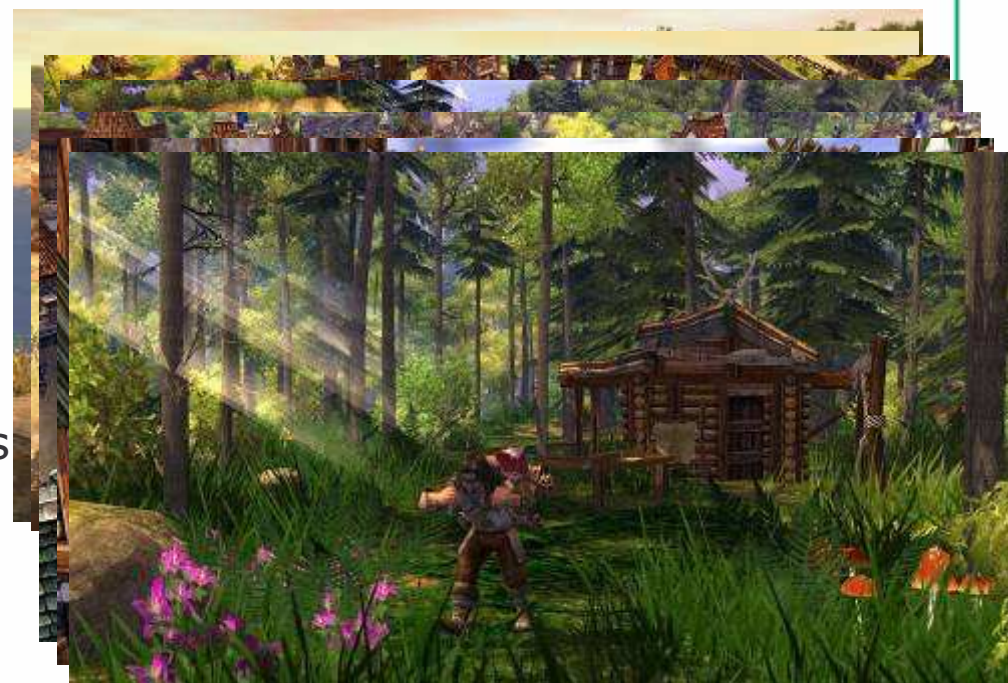
- Start profiling early
 - Learn bad news early
 - Always be aware of impact of new code

Agenda

- Introduction
- Motivation
- Introduction to AMD GPUs Tools
- Preconditions for profiling the GPU load of a game
- Case Study: Blue Bytes "The Settlers – 6" (working title)
- Conclusions
- Questions and Answers

Case Study: Settlers 6 (working title)

- Developer is BlueByte
 - Development started spring '05
 - Will be released this summer
- Empire building RTS game
 - Set in a medieval world
 - Has a focus on city building
 - Warfare part of the game
- Single & Multi player modes
 - Various multi player modes
 - Story based and free-play maps
- Complex graphics engine
 - Highly dynamic world
 - Lots of foliage



The Settlers®

©2006 Ubisoft Entertainment. All Rights Reserved. The Settlers, Blue Byte and the Blue Byte logo are trademarks of Red Storm Entertainment in the US and/or other countries. Ubisoft and the Ubisoft logo are trademarks of Ubisoft Entertainment in the US and/or other countries. Red Storm Entertainment Inc is a Ubisoft Entertainment company. Developed by Blue Byte Software.

Case Study: Settlers 6 (working title)



The Settlers®

©2006 Ubisoft Entertainment. All Rights Reserved. The Settlers, Blue Byte and the Blue Byte logo are trademarks of Red Storm Entertainment in the US and/or other countries. Ubisoft and the Ubisoft logo are trademarks of Ubisoft Entertainment in the US and/or other countries. Red Storm Entertainment Inc is a Ubisoft Entertainment company. Developed by Blue Byte Software.

Decision to use the tools was easy

- Tools are not invasive
 - No changes to your source code
 - No additional risk even late during a project
 - No additional Q&A 😊
- Tools would allow most efficient way to ...
 - Assess impact of rendering techniques for new scene elements
 - Only one scene element covered by this talk

Initial Situation

- Graphics engine was already very mature
 - Well optimized
 - Deals with highly dynamic landscapes
 - Existing support for enabling/disabling scene elements ☺
- Additional small scene elements were about to be added
 - More grass
 - Flowers, Pebbles, Toadstools etc.
- We will talk about rendering flowers
 - Representative for all small scene details listed above

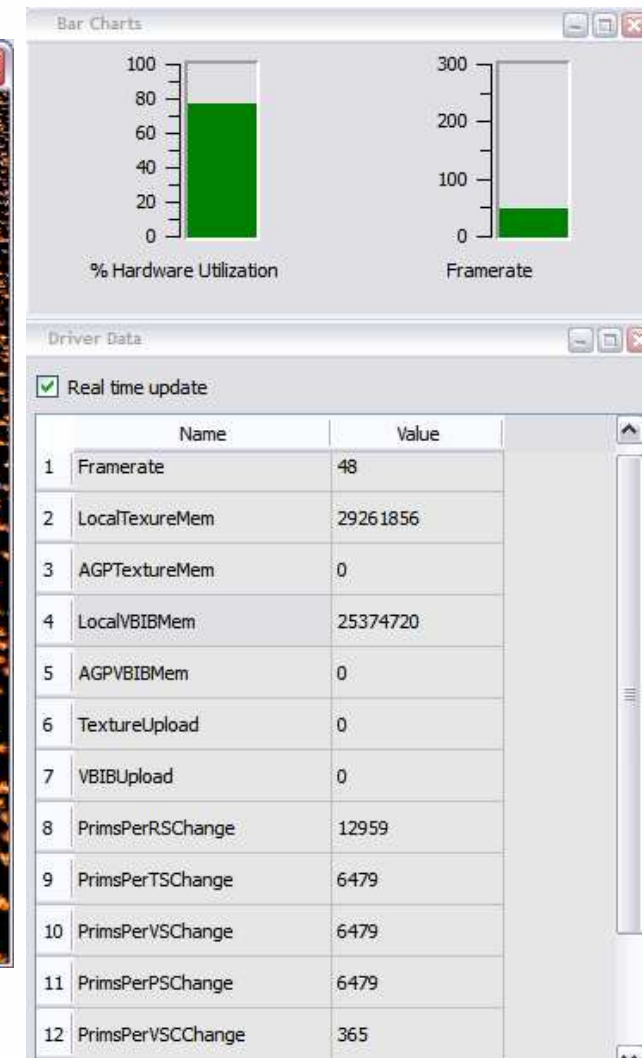
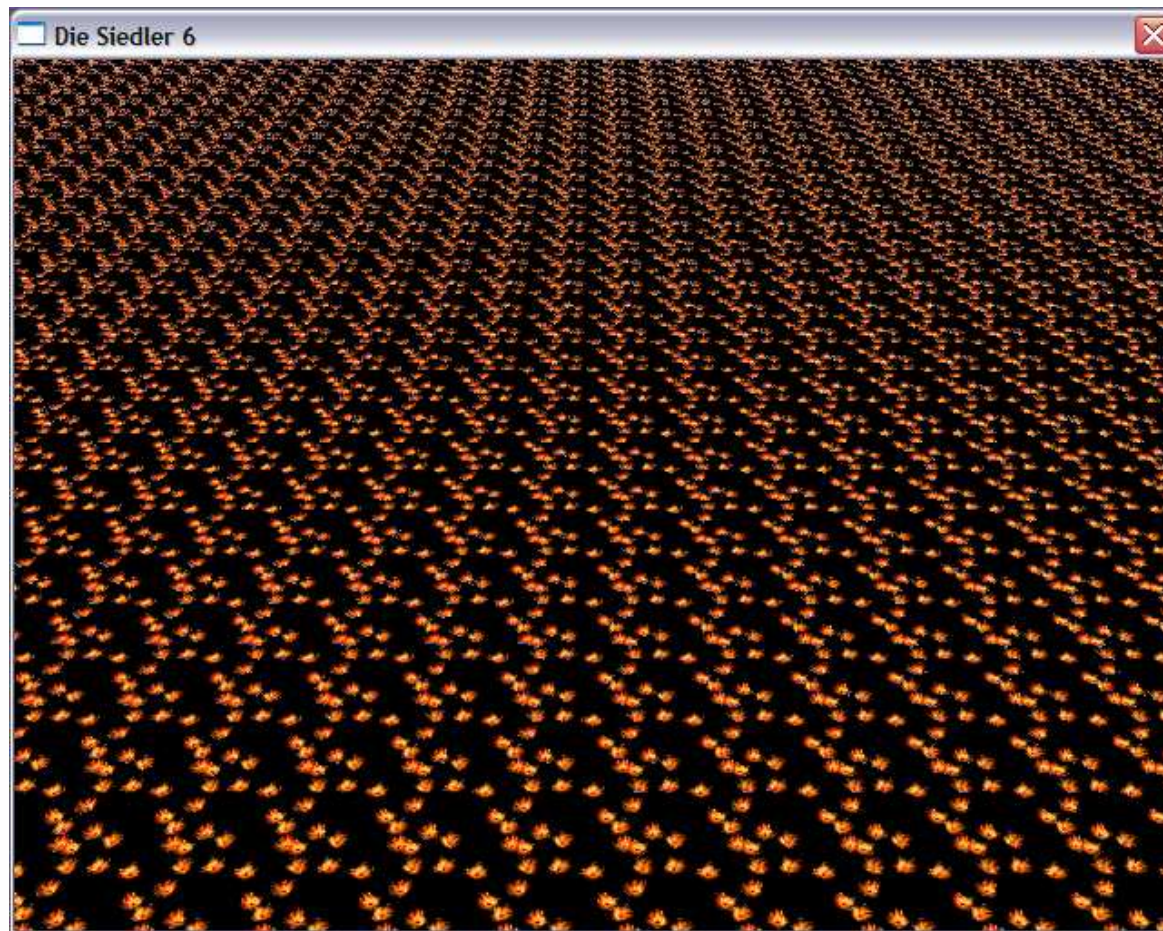
Initial implementation of Flowers

- Requirements:
 - Batches of flowers small enough to adapt to terrain
 - Don't want to render just one flower in a draw call
- Implementation:
 - render a bunch of flowers per draw call
 - It is known that the VS is relatively expensive
 - Scattering is done inside VS
- Let's use GPU PerfStudio to look at that implementation

Before we really start...

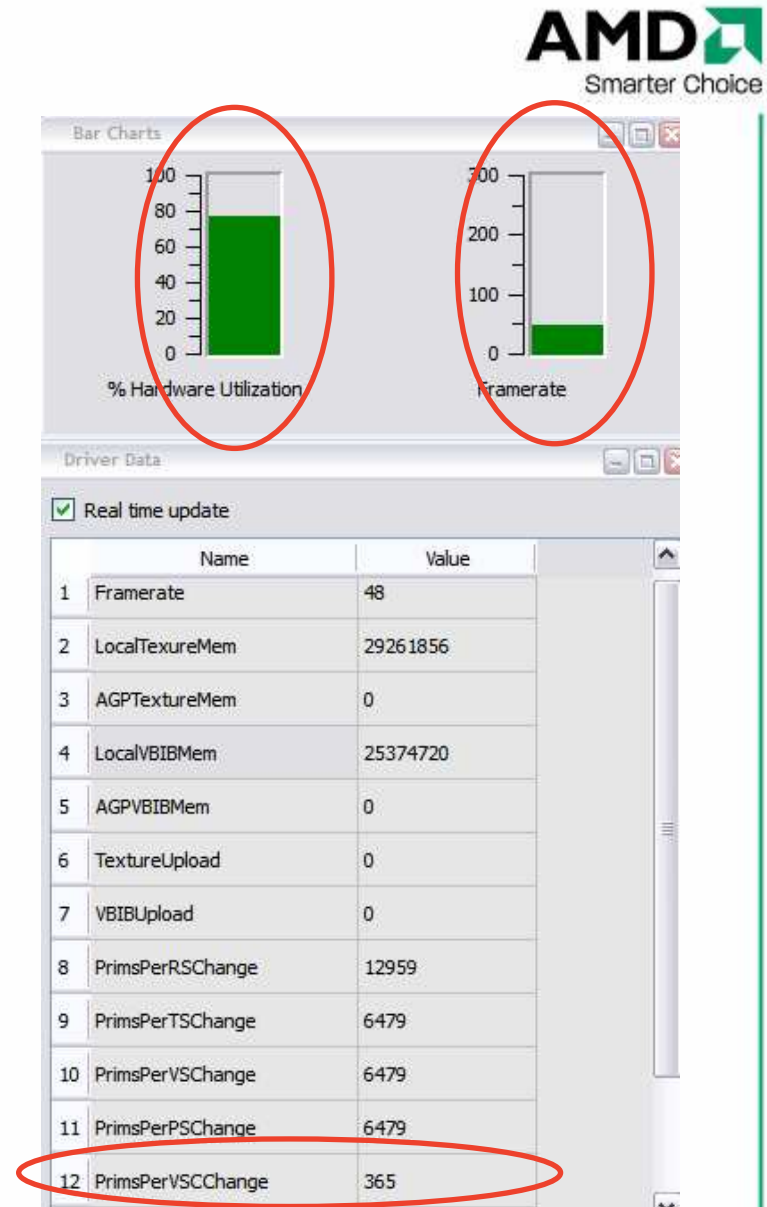
- Talk has been created around results on this machine
 - X1300 laptop accelerator
 - Results will vary on desktop cards or other laptops
 - Will pretend that this laptop is a constant target platform
 - With constant resolution, memory bandwidth etc.
- Wanted results that can be repeated during the talk
- Won't talk about every feature of the tools
- Concentrate on easy to understand measurements
- Case study is about a DX9 title
 - supposed to run on DX9 hardware
 - Not on unified shader hardware

Let's run GPU PerfStudio ...



Results - Flowers

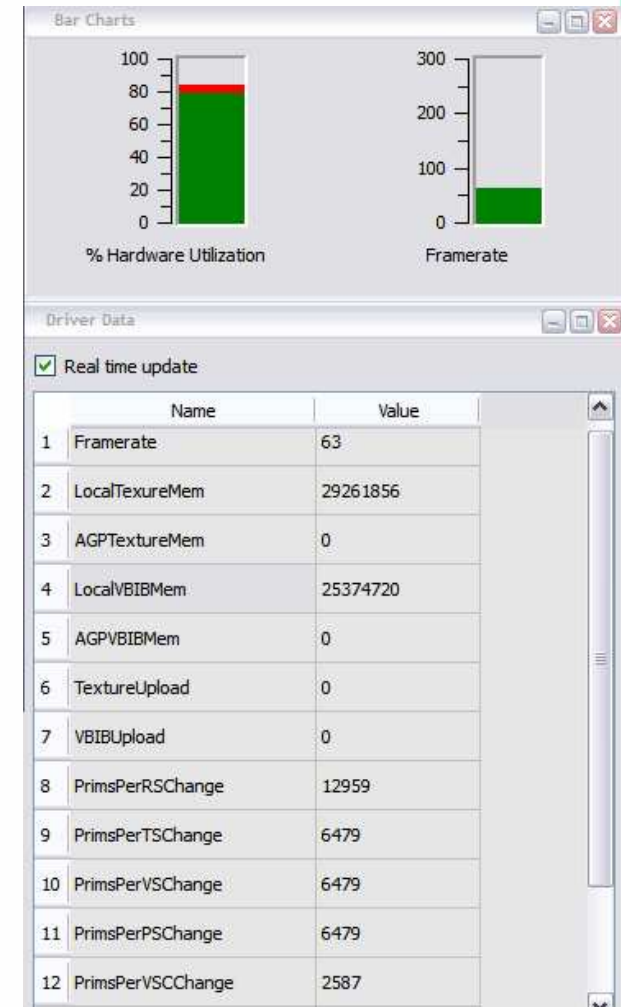
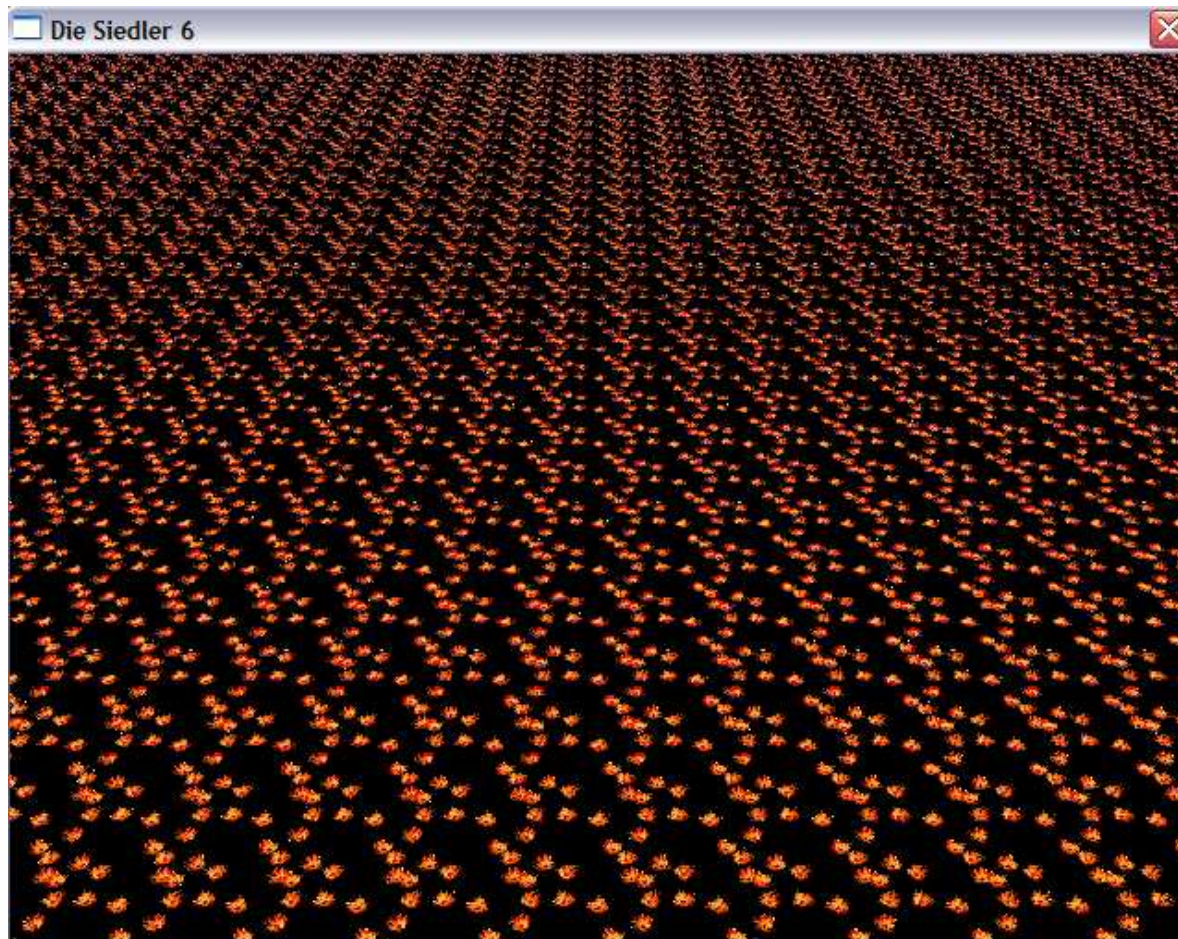
- Hardware utilization not bad
- FPS ~50 ... not too good
- But we want to maximize FPS
 - And we can do better !
- PrimsPerVSCChange is low !



Improving the performance

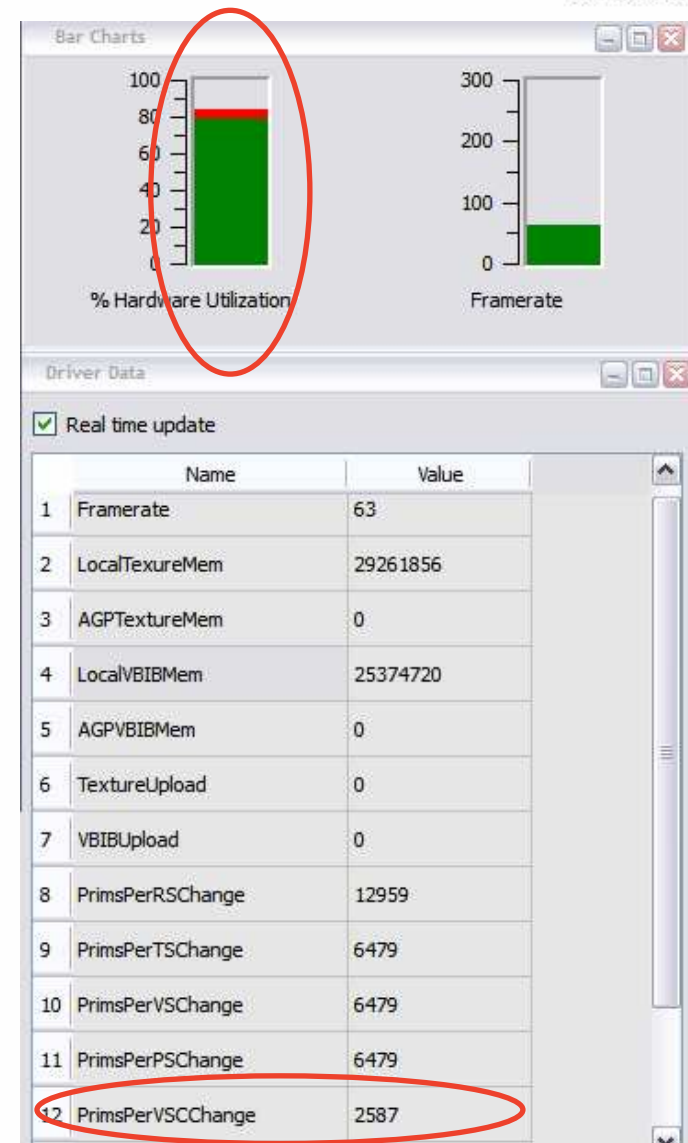
- We'll use instancing to increase PrimsPerVSCChange
- Our options are:
 - Hardware instancing
 - Shader constants instancing
- Shader constants instancing used for max compatibility
- VertexBuffer contains several copies of geometry
- Vertices contain index to constants array of positions
- Now we can draw several batches of flowers at once

Let's run GPU PerfStudio again ...

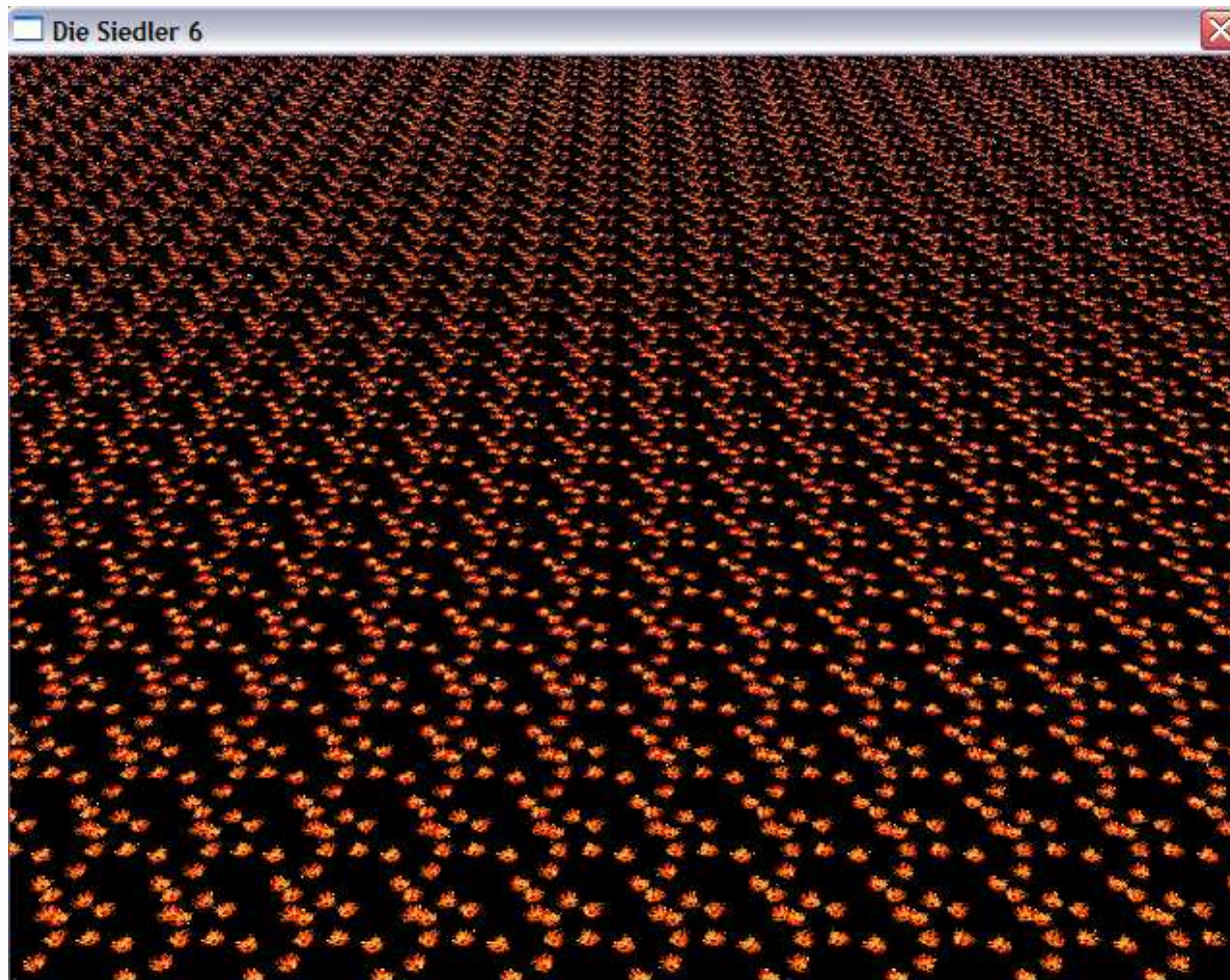


Improved Results – Instanced Flowers

- Hardware better now
- PrimsPerVSCChange went up a lot
- Fps ~60 !
- We are happy now ! 😊
- Can do even better ?



Let's run GPU PerfStudio again ...



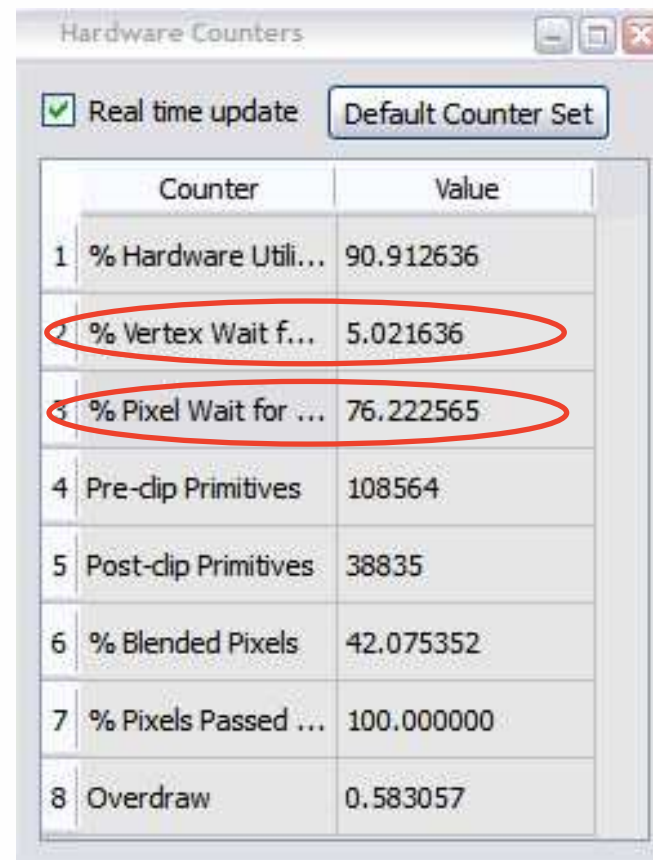
Hardware Counters

☒ Real time update Default Counter Set

	Counter	Value
1	% Hardware Utili...	90.912636
2	% Vertex Wait f...	5.021636
3	% Pixel Wait for ...	76.222565
4	Pre-clip Primitives	108564
5	Post-clip Primitives	38835
6	% Blended Pixels	42.075352
7	% Pixels Passed ...	100.000000
8	Overdraw	0.583057

What else is not optimal ?

- Let's look at vertex vs. pixel loads
- % Vertex wf Pixel ~ 5
- % Pixel wf Vertex ~ 75
- This indicates a vertex limitation
 - We know it is not vertex locality!
 - Vertex locality has been optimized



	Counter	Value
1	% Hardware Util...	90.912636
2	% Vertex Wait f...	5.021636
3	% Pixel Wait for ...	76.222565
4	Pre-clip Primitives	108564
5	Post-clip Primitives	38835
6	% Blended Pixels	42.075352
7	% Pixels Passed ...	100.000000
8	Overdraw	0.583057

How can we still improve our frame rate on this machine?



- We could move ALU load from VS to PS
 - This is no general advice!
 - only makes sense in this context on this constant target machine
 - With this fixed window resolution etc.
 - Other machines may behave differently
 - Different rules may apply for DX10 or unified shader HW!
- Let's first assess the current pixel shader
- Let's start GPU ShaderAnalyser

Running GPU ShaderAnalyzer

- For the case study shaders were optimized for X1900
 - Will look at results for X1900
 - Of course X1300 would have a different ratio
- Ratios tell us PS can abide for more ALU on X1900
 - Especially for Trilinear and Aniso
- Let's move scattering computations from VS to PS
 - Won't go into details about the shader changes
- And let's start GPU ShaderAnalyzer again

GPU	Vertex	Fragment	Texture	ALU	Texture	ALU	Texture	ALU	Texture
Radeon x1800 (R520)	7	18.00	18.00	18.00	2.60	3.00	2.57	N/A	N/A
Radeon x1900 (R580)	7	7.00	7.00	7.00	1.40	1.17	1.00	N/A	N/A

D3D Assembly Statistics

Shader Version = 3.0
Instruction Count = 120
ALU Instructions = 81, Texture Instructions = 15, ALU:Texture Ratio = 5.40

Constant Register Count = 4569
Temp Register Count = 20, Sampler Register Count = 10, Input Register Count = 9, Output Register Count = 3

Requires PS3.0

Reassessing the situation

- Ratios went up for X1900
 - It is what we would expect ☺
- We are now green even for Aniso
- Let's check what change we get on this machine
 - Let's start GPU PerfStudio again

GPU ShaderAnalyzer - OrnamentalItemPSScatter.fx

Source Code: Function: main

HLSL Compiler: Compile Options

Object Code: Format: Radeon x1900 / PSS

```
94  
95     return ColorWithScattering(Color, Extinction, float4( Scattering, 0.0f));  
96 }  
97
```

Compiler Statistics (Using Catalyst 7.1)

ASIC	Registers	Cycles (Bilinear)	Cycles (Trilinear)	Cycles (Aniso)	ALU:Tex (Bilinear)	ALU:Tex (Trilinear)	ALU:Tex (Aniso)	SCDev Registers	SVDev Insts
Radeon x850 (R480)	9	47.00	47.00	47.00	9.40	7.83	6.71	N/A	N/A
Radeon x1800 (R520)	8	31.00	31.00	31.00	6.20	5.17	4.43	N/A	N/A
Radeon x1900 (R580)	8	11.00	11.00	11.00	2.20	1.83	1.57	N/A	N/A

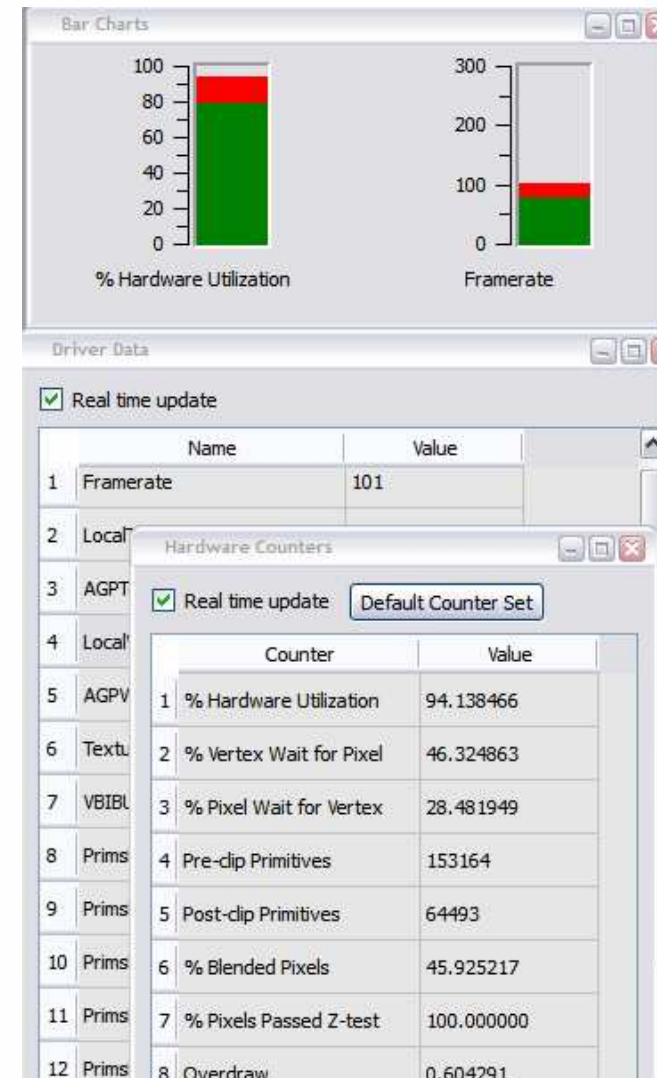
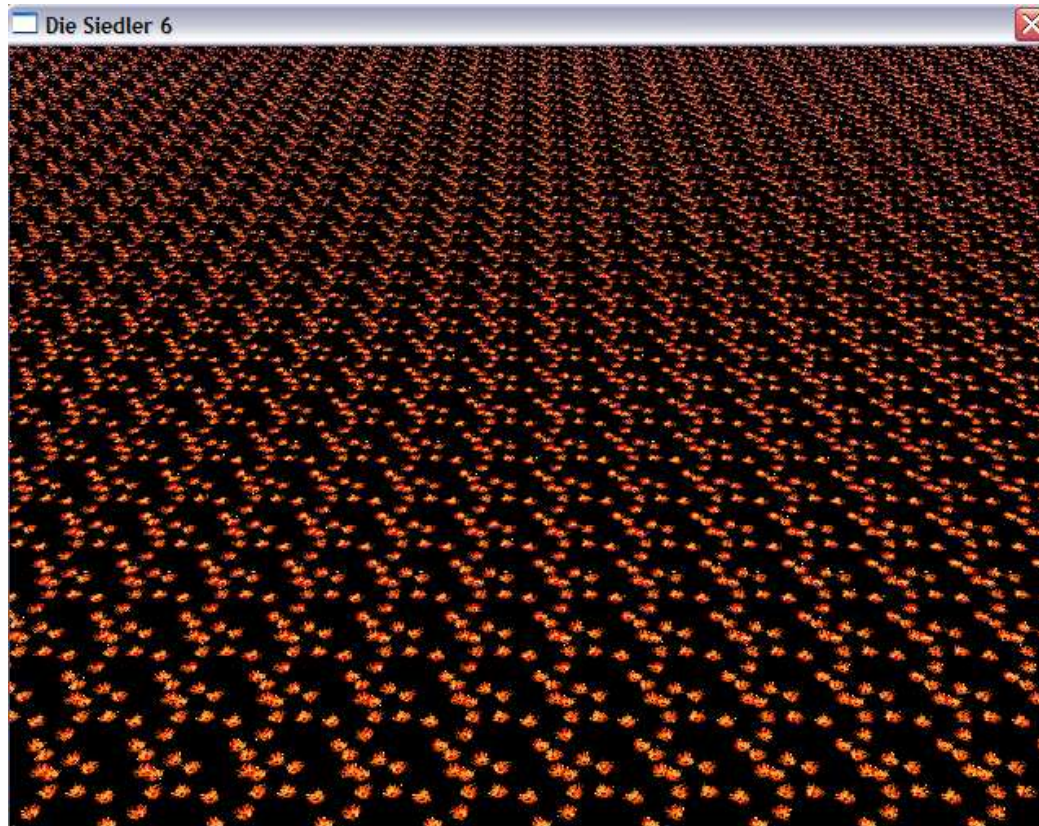
D3D Assembly Statistics

Shader Version = 3.0
Instruction Count = 195
ALU Instructions = 142, Texture Instructions = 20, ALU:Texture Ratio = 7.10

Constant Register Count = 4740
Temp Register Count = 20, Sampler Register Count = 16, Input Register Count = 13, Output Register Count = 5

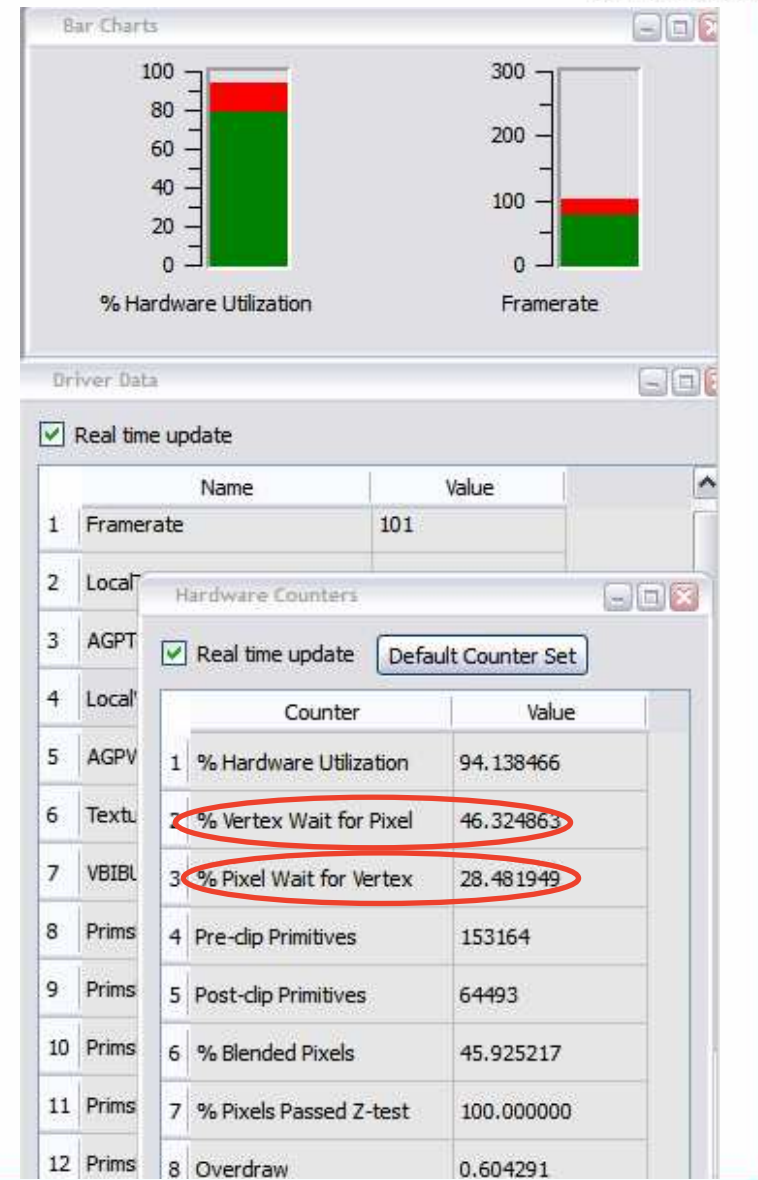
Requires PS3.0

Let's run GPU PerfStudio again ...



Observation & Results

- Reversed Pixel vs. Vertex
 - We're pixel limited now
- This is not necessarily a problem
 - It is hard to get balance
- FPS has increased
 - Now ~100 FPS
- We did the right thing 😊
 - At least on this machine
 - And this window resolution



Agenda

- Introduction
- Motivation
- Introduction to AMD GPUs Tools
- Preconditions for profiling the GPU load of a game
- Case Study: Blue Bytes "The Settlers – 6" (working title)
- Conclusions
- Questions and Answers

Conclusions & Call to action

- AMD GPU tools are extremely valuable tools
 - They allow you to look inside the GPU
 - Find reasons for performance bottlenecks
 - Help to remove performance bottlenecks
- AMD GPU tools are free
- Try them as soon as possible
- Start to understand your GPU 😊

Agenda

- Introduction
- Motivation
- Introduction to AMD GPUs Tools
- Preconditions for profiling the GPU load of a game
- Case Study: Blue Bytes "The Settlers – 6" (working title)
- Conclusions
- Questions and Answers

Thank you for listening ...



Q&A

Holger Gruen
European Developer Relations
Holger.Gruen@amd.com